AMENDMENTS TO THE SPECIFICATION

Please amend the paragraph beginning on page 2, line 8 and ending on page 3, line 5 as follows:

An example of a simplified instruction pipeline 100 is shown in Figure 1. According to this simplified example, the instruction pipeline 100 comprises five major stages ~~105-130~~105-125. The five major stages are the fetch stage 105, the decode stage 110, the dispatch stage 115, the execute stage 120, and the writeback stage (also referred to as the retirement stage) 125. Briefly, during the first stage, the fetch stage 105, one or more instructions are retrieved from memory, and subsequently decoded into micro-ops during the decode stage 110. Then, the micro-ops are dispatched to the appropriate execution unit for execution during the dispatch stage 115 and execution takes place during the execute stage 120. Finally, as the micro-ops complete execution, they are marked as being ready for retirement and are subsequently retired (e.g., their results are committed to the architectural registers) during the retirement stage 125. Consequently, the fetch unit (not shown) at the head of the pipeline provides the pipeline with a continuous flow of instructions, hence keeping the microprocessor busy. The fetch unit keeps the constant flow of instructions so the microprocessor does not have to stop its execution to fetch an instruction from memory. Such fetching guarantees continuous execution, as long as the instructions are stored in order of execution. However, due to

branch instructions, such as conditional branch instructions included in software loops or conditional jumps, instructions encountered by the fetch unit are not always presented in a sequence corresponding to the order of execution. Thus, branch instructions can cause pipelined microprocessors to speculatively execute down the wrong path such that the microprocessor must later flush the speculatively executed instructions and restart at a corrected address.

Please amend the paragraph beginning on page 3, line 20 and ending on page 4, line 6 as follows:

An earlier branch target buffer cache implementation is illustrated in Figures 2 and 3. The branch target buffer (BTB) 200 depicted in Figure 2 is a set-associative cache that stores information about branch instructions in 128 individual "lines" of branch information. Each line of branch information in the BTB 200 contains four branch entries that each contains information about a single branch instruction that the microprocessor has previously executed (if the valid bit is set in the entry). Each line also includes a branch pattern table 221 and least recently replaced (LRR) bits 220. The branch pattern table 221 is used for predicting the outcome of conditional branch instructions in the line of branch entries. The [[LLR]] LRR bits 220 are used by the branch prediction circuit to select a branch entry in the line when information about a new branch will be written into the line of branch entries.

Please amend the paragraph beginning on page 12, line 21 and ending on page 13, line 7 as follows:

Branch Prediction Circuit

Figure 5 is a simplified block diagram of a branch prediction circuit 500 according to one embodiment of the present invention. In the embodiment depicted, the branch prediction circuit 500 includes an architectural branch target buffer (ABTB) 510, a speculative branch target buffer (SBTB) 520, and selection logic 530. According to one embodiment, the SBTB 520 is a relatively small structure supporting the ABTB 510. The SBTB 520 is used to maintain the speculative branch data for in-flight branches, meaning fetched branches that are not yet retired. In the embodiment depicted, the SBTB includes an N stage FIFO, where N is the number of stages in the processor's instruction pipeline and a branch allocation register-[[523]] 553. Each stage of the FIFO includes per-line fields 521 and per-way fields 522.

Please amend the paragraph beginning on page 14, line 21 and ending on page 15, line 9 as follows:

Branch Entry Processing

Figure 6is a flow diagram illustrating branch entry processing according to one embodiment of the present invention. When no entry is found in the ABTB 601, an entry is allocated in the SBTB at decode time 603. When an entry is found in the ABTB, a prediction is taken at 602. In case of a bogus branch 604, deallocation is performed at decode time 605, else the branch is predicted speculatively 606. The speculative prediction continuous 606, assuming the prediction is correct, for the subsequent entries

until an actual entry in found in the ABTB 607. Once there is an actual entry in the ABTB, any corresponding entry in the SBTB is decoded in order to avoid duplication 608. If no actual entry is found in the ABTB, speculative prediction continues at 609. Any mispredicted branches and mispredicted targets are corrected at execute time, and entries are later executed 610. Finally, the executed branch instructions are retired 611. The branch history and PT are updated, but only branches that actually retire update the ABTB at 612. Since not all executed branches retire, branch update at Retire time eliminates corruption.